Improving Performance of Scientific Programs with OpenACC.

Frank Kornet HCC, Houston, TX (frkornet@gmail.com)
Mentor: Millad Ghane, University of Houston, Houston, TX (millad.mg@gmail.com)

NSF Grant: PHY-1427654

Parallel Programming

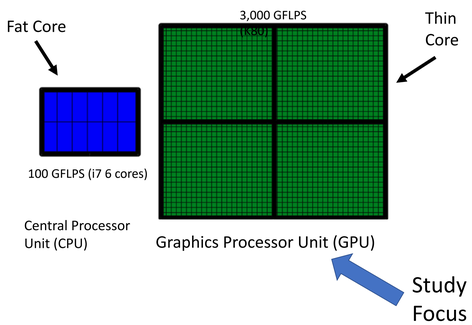
```
#pragma acc parallel loop
for (int i=nLocalBoxes;
     i < nTotalBoxes;
     ++i)
     s_boxes_nAtoms[i] = 0;
```

Example of OpenACC compiler directive.

Why Parallel Programs?

- Moore's law running out of steam and scientific applications will no longer get an automatic performance boost
- Scientific applications need to run ever larger simulations. To do so, they need to become faster
- The only realistic way to increase performance is by writing parallel programs

CPU and GPU Cores



CUDA vs OpenACC

	CUDA	OpenACC
Platforms	NVIDIA GPUs only	CPUs and GPUs
Languages	Fortran and C/C++	Fortran and C/C++
Usage	Function call	Compiler directive
Level	Lower	Higher

Hypothesis:

- OpenACC performs at similar levels than CUDA
- OpenACC easier to use and more productive than CUDA

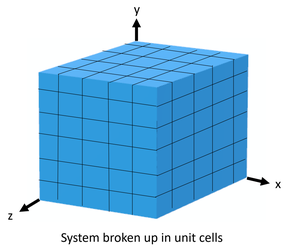
Objectives

- Demonstrate that OpenACC's performance is within 10 – 20 per cent of CUDA's performance
- Demonstrate that OpenACC is easier to use and more productive than CUDA
- Personal: learn parallel programming concepts and physical concepts behind the Verlet algorithm

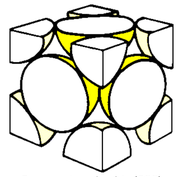
Use CoMD as Proxy

Molecular dynamics (MD) is the method of simulating the kinetic and thermodynamic properties of molecular systems using Newton's equations of motion and represents an important class of scientific applications. CoMD is used as it is both small and representative of this class. It carries out a microcanonical ensemble simulation (NVE) and is developed by DoE Co-design Center for Materials in Extreme Environments (ExMatEx).

Los Alamos NATIONAL LABORATORY · Lawrence Livermore National Laboratory · OAK RIDGE National Laboratory · Sandia National Laboratories · STANFORD UNIVERSITY



System broken up in unit cells



Face-centered cubic (FCC)

Copper periodic table

Number of Atoms
4 * X * Y * Z

Force Calculation
Neighbor Cells

Force Model
Lennard-Jones & EAM (pair-wise with cutoff)

Sample Screen Output

Constant · Variable · Constant

```
# Loop Time(fs) Total Energy Potential Energy Kinetic Energy Temperature Performance # Atoms
     0     0.00  -3.46052323      -3.53807922     0.07755599    600.0000      0.0000   32000
    10    10.00  -3.46052263      -3.52992415     0.06940151    536.9142      0.9557   32000
```

Total Energy = Potential Energy + Kinetic Energy

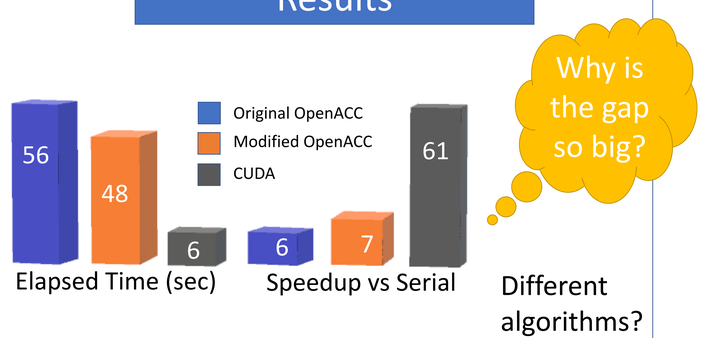
CoMD implements an $O(N^2)$ Verlet algorithm. For large simulations, CoMD will split the system and distribute cells over participating nodes. A partial OpenACC version and a full CUDA version are available on GitHub and are used in the study.

Change Made

Function	CPU	GPU
main	X	
timestep	X	
sumAtoms		X
printThings	X	
advanceVelocity (*)		X
advancePositions (*)		X
redistributeAtoms	X	
updateLinkCells	X	
haloExchange	X	
sortAtomsInCell		X
computeForce (*)		X
advanceVelocity (*)		X
kineticEnergy		X

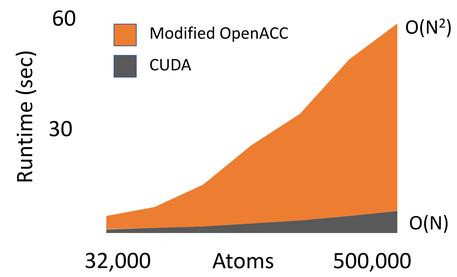
(*) Basic Momentum Verlet algorithm (2nd version) as described in http://www.chem.utoronto.ca/~jmschofi/simulation/partmd.pdf

Results



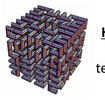
Force model: EAM. Simulating 500,000 atoms (100 steps) on single node with 2 x K80.

Algorithmic Behavior



Force model: Embedded Atom Model (EAM)
Simulating 100 steps on single node with 2 x K80.

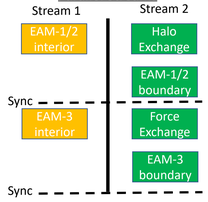
CUDA Modifications



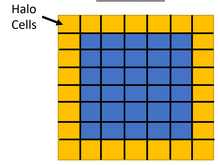
Hilbert Curve to improve texture caching

Neighbor Atoms w/in cutoff

Parallel Streams



Sort Atoms



- Only sort atoms in Halo Cells
- Optimize sort (hashing)

Source: http://on-demand.gputechconf.com/gtc/2014/presentations/S4465-optimizing-comd-molecular-dynamics.pdf

Conclusions

- OpenACC vs CUDA is not a like for like comparison. It is unclear, though, which of the changes result in the different algorithmic behavior
- OpenACC is easier to use and needs less code than CUDA, but different algorithm overstates CUDA part

	OpenACC Original	OpenACC Modified	CUDA
Lines of Code	6,200	6,900	13,000

- Next step: migrate haloExchange to GPU and investigate what truly drives the CUDA algorithm
- Personal objectives:

OpenACC	Verlet	Physics
√	√	50/50

- I enjoyed learning about physics!